Visualizing Weighted Edges in Graphs

Peter Rodgers and Paul Mutton
University of Kent, UK
P.J.Rodgers@kent.ac.uk, pjm2@kent.ac.uk

Abstract

This paper introduces a new edge length heuristic that finds a graph layout where the edge lengths are proportional to the weights on the graph edges. The heuristic can be used in combination with the spring embedder to produce a compromise between a drawing with an accurate presentation of edge length and a drawing with good general comprehensibility. We describe our preliminary investigations in combining the two methods so that a user can tune their preference and demonstrate the effectiveness of the system on both randomly generated graphs and graphs representing web page similarity data.

Keywords: *graph drawing, metric embedding.*

1: Introduction

The automatic layout of interconnected data, termed graph drawing, is widespread in information visualization. In many application areas, the graphs being visualized are weighted, where the weights on the edges represent a measure between connected nodes. Examples of such data are web page similarity graphs or the graphs depicting the closeness of bibliography entries. For visualizing this sort of information, an important requirement is that the displayed edge length should be proportional to the weight. However, a good solution for edge weights is often at the cost of making the overall graph drawing less comprehensible, so the user finds that examining the data is difficult.

A widely used method for drawing graphs with proportional edge lengths is to apply a metric spring embedder to the graph. This applies the spring embedder, first developed by Eades [1], with modifications made to ensure that the attractive forces acting on edges are inversely proportional to the edge weight. Many applications use this approach, such as visualizations for biological data [3] or showing the similarity between news stories [2]. Tools to support this form of graph drawing have been developed, including NetVis ViStA [5], and some force directed approaches are easily modified to optimize on edge length [4]. The result of using these methods is that the graph is often nicely drawn for comprehensibility and that the edge weights and edge lengths have some relationship. However, the resultant edge lengths are usually far from their ideal values, even when the graph has a possibly exact solution.

Metric embedding techniques, which attempt to optimise laying out edge lengths proportional to their weights have been widely applied. Finding an optimal solution for this problem is NP-Complete and so such techniques are approximative. They include multidimensional scaling methods and hierarchical decomposition [7]. The result of such techniques is good for edge lengths, but the layout is also usually a jumbled graph that is difficult to analyse visually.

In this paper, we introduce a new edge length method that can be closely integrated with the metric spring embedder to produce a compromise between a comprehensible graph and a metric embedding.

As with the spring embedder, the system iterates a number of times until a reasonable drawing has been found. On every iteration, the metric spring embedder method is first applied to the nodes in the graph, and then the edge length heuristic is applied to the nodes in the graph. No modification is required to the calculation of the forces in the spring embedder. On any iteration, it is possible to apply either or both methods, so providing a tuning mechanism where the user can decide how much spring embedding or edge length method they require for their final drawing depending on their emphasis on comprehensibility or edge length proportionality. The edge length method is more powerful than the spring embedder and produces an equilibrium faster. Hence, to produce a more understandable graph it can be left out for several iterations whilst the spring embedder is applied on every iteration. This means the tuning is defined as how frequently the edge length method is applied.

To apply an iteration of the edge length heuristic, the method iterates through the nodes in the graph. For each node, the edge length heuristic takes each connecting edge and finds the ideal length based on multiplying the weight of the edge by a constant. The position of the node to create that ideal length is then calculated. Once all the positions for the connecting edges are calculated, the node is then moved to the average of the positions.

The heuristic can be implemented with $O(|E|)$ time complexity for each iteration, where $|E|$ is the number of edges in the graph. This compares well with the spring embedder which is $O(|N|^2)$ for each iteration of the classical embedder [1], where $|N|$ is the number of nodes in the graph, although $O(|N| \log |N|)$ can be achieved with optimisations. Hence, the scalability of our method is largely dependent on the spring embedder rather than the heuristic. The spring embedder has been applied to graphs of over a hundred thousand nodes [8].

A version of the edge length heuristic was first developed as a preprocessor for the standard spring embedder [6]. Rather than making edge lengths proportional to weight, it was designed to make edge lengths equal. In this form it was combined with a grid allocation method in a two phase system. It has the effect of evening out edge lengths and giving nodes a minimum separation, so approximating the goal of the spring embedder. This was shown to significantly reduce the time taken for subsequent spring embedding. This investigation also indicated that graph drawings produced by using the heuristic alone were often unclear and that good results for edge length are at the expense of providing users with a generally comprehensible diagram.

The remainder of this paper is organized as follows: Section 2 describes the edge length heuristic in more detail; Section 3 gives the results of using the method in various combinations with the metric spring embedder on test data; and finally, Section 4 gives our conclusions and directions for further research.

2: Description of the Method

This section describes the edge length method in detail and discusses how it is integrated with the spring embedder to produce a tuneable graph drawing system.

For our experiments, we consider connected graphs in two-dimensional space. We define a graph G = (N,E), where N is the set of nodes and E is the set of edges that connect nodes together. We also define a weight function, W, assigning a real number to all the edges in E.

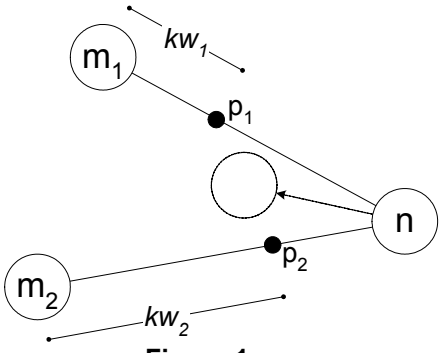

Figure 1

A diagrammatic representation of the effect of the edge length heuristic on Node n.

The edge length heuristic places a node n at the average of the ideal positions for each connecting edge. It repeats this for each node in the graph. For each edge e_i with weight w_i that connects node n to node m_i, we calculate the position of point p_i such that the vector $m_ip_i = kw_i(u_i)$, where u_i is the unit vector of m_in, see Figure 1. This can be considered the ideal location of the node n if it connected to only the node m_i. k is a constant that adjusts the final edge length according to the desired edge length and the values of the weights, so that k can be reduced if the weights in a particular graph are large. The node is then moved to the average of all the locations p_i.

The edge length heuristic in pseudocode:

```
APPLY_EDGE_LENGTH(G)
  FOREACH n IN G
    F = setOfConnectingEdges(n)
    P = {}
    FOREACH e_i IN F
      Calculate location of p_i
      Add p_i to P
    Move node to mean location in P
```

The metric spring embedder method operates like a standard classic spring embedder, see [1] except that on each iteration, the repulsive force calculation for each edge e_i is multiplied by the inverse of w_i.

The two methods are then combined in the graph drawing system, which iterates a user defined number of times. On each iteration, both, or just one of the methods can be applied. In general, the spring embedder is always applied, except in the case where the edge length method only is required. The edge length method can be applied on every iteration, or less frequently, such as every 5 iterations. Where the spring embedder only is required, the edge length method is not applied at all.

The graph drawing method in pseudocode:

```
DRAW(G)
FOREACH iteration, i
  if(USE_SPRING_EMBEDDER(i))
    APPLY_SPRING_EMBEDDER(G)
  if(USE_EDGE_LENGTH(i))
    APPLY_EDGE_LENGTH(G)
```

3: Results

In order to gauge the effect of applying the edge length heuristic, we performed four graph drawing variations on a set of test graphs. The variations were various ways of combining the edge length method with the metric spring embedder. The idea was to apply a different emphasis on the two conflicting goals of producing comprehensible graphs and making the edge lengths in the graph as proportional to the edge weights as possible. The four variations were: spring embedding only (SE); spring embedding every iteration and edge length every five iterations (SE5EL1); both spring embedding and edge length on each iteration (SE1EL1); and edge length method alone (EL).

The test graphs, listed in Figure 2, are a combination of randomly generated graphs (those graph names starting with ‘random’) and graphs generated from web pages (those graph names starting with ‘www.’), with similarity data where nodes represent web pages, edges represent hyperlinks between pages, and edge weights are values representing the number of shared links of the two pages. In addition there are a few hand created graphs to explore specific possible issues (those graphs named ‘metric’, ‘nonmetric’, ‘general’, ‘simple’ and ‘clustered’) and a graph derived from real world data (the graph named ‘bibliography’, see [2]).

Graph	Number of Nodes	Number of Edges
metric	3	3
nonmetric	3	3
random10-20	10	18
thing	17	30
simple	18	30
www.web-bits.net	18	51
clustered	19	30
random20-40	20	38
www.brettmeyers.com	23	60
www.counter-strike.net	25	26
random30-60	28	58
www.a-spotted-dog.com	31	129
random40-80	40	77
random50-50	43	49
random50-75	47	72
random50-100	49	97
random50-150	50	149
random50-200	50	195
random50-300	50	297
random60-120	60	115
random80-160	80	156
random100-200	98	196
www.bersirc.com	104	803
www.ivarjohnson.com	111	225
random150-300	150	297
random200-400	194	399
www.peacenikjive.com	226	669
random300-600	296	598
random400-800	390	799
random500-1000	496	998
bibliography	504	754

Figure 2
The test graphs.

Our criteria of success for the edge length accuracy was a calculation of distortion, a measure of how close the edge lengths in the graph are to their ideal. The distortion we used is given by:

$$1.0E6 \sum_{edge\ e} \frac{\left(l(e) - w(e) \times ul(e)\right)^2}{|E|} \Bigg/ \left(\sum_{edge\ e} l(e) \right)^2$$

where *l(e)* is the length edge *e*, *w(e)* is the weight of edge *e*, *ul(e)* is the desired unit length if edge *e*, i.e. the sum of edge length in the graph divided by the number of edges. This allows us to multiply *ul(e)* by the weight of an edge *w(e)* to give us the ideal weight for *e*. We divide by the total edge length squared to make the measure dimensionless, that is, the measure does not vary on the physical size of the graph, only on the relative differences of the edge lengths. The constant 1.0E6 has been factored in so that the numbers are of a magnitude that makes for easier comparisons.

Graph	SE	SE5EL1	SE1EL1	EL
metric	2069.17	1.06	0.16	0.18
nonmetric	29821.14	20023.50	20025.33	20022.71
random10-20	526.00	184.82	190.89	195.22
thing	195.39	210.86	33.29	4.25
simple	79.30	56.25	17.78	0.13
www.web-bits.net	24.44	12.75	11.06	11.34
clustered	179.27	17.50	8.14	8.50
random20-40	115.49	90.10	53.37	27.47
www.brettmeyers.com	13.06	3.31	1.53	1.12
www.counter-strike.net	59.66	0.18	0.03	0.00
random30-60	42.81	35.50	15.26	7.75
www.a-spotted-dog.com	8.92	1.93	1.54	1.49
random40-80	25.54	3.62	1.26	1.01
random50-50	63.14	35.52	8.23	3.62
random50-75	29.14	3.37	1.78	1.53
random50-100	18.63	3.32	2.08	1.97
random50-150	9.55	2.83	2.13	2.25
random50-200	5.42	1.88	1.63	1.68
random50-300	2.70	1.09	1.02	1.00
random60-120	14.06	3.10	1.54	1.46
random80-160	7.58	1.49	0.59	0.51
random100-200	4.65	1.10	0.38	0.25
www.bersirc.com	0.39	0.04	0.02	0.02
www.ivarjohnson.com	3.29	1.75	0.36	0.07
random150-300	1.84	0.58	0.21	0.11
random200-400	1.08	0.42	0.11	0.03
www.peacenikjive.com	0.63	0.25	0.10	0.07
random300-600	0.49	0.24	0.06	0.01
random400-800	0.29	0.15	0.05	0.01
random500-1000	0.18	0.11	0.03	0.00
bibliography	0.47	0.39	0.17	0.01

Figure 3
The amount of distortion. The values with a grey background are those which are higher than the value in the column to the immediate left.

Each graph drawing variation was applied to each graph with approximately 1000 iterations used. Before each application, the positions of the nodes were randomized before the variation was applied. The number of iterations was set so that the SE5EL1 variation finished on 4 spring embedding iterations, rather than the iteration that included both methods. Some modifications to values for the edge length and spring embedder methods were made to ensure reasonable results. The measures were then applied to the resultant layouts. This was repeated 10 times. The average distortion of these experiments is shown in Figure 3.

From Figure 3, it can be seen that EL variation is by far the most effective in reducing distortion. The SE1EL1 variation is consistently the next most effective. Using the spring embedder every iteration and the edge length method every 5 iterations is more effective than using the spring embedder alone. The data points that do not bear out these conclusions are shown with a grey background,

and are particularly prevalent in smaller graphs. Most of these points have very small differences Some of these figures are because the node coordinates have integer values, and so some rounding occurs. This is certainly true in the case of "metric" and "nonmetric". The only significant case where the negative case does not involve the EL variation is in "thing", where the SE1EL1 is surprisingly worse than the SE alone. Its not clear why this should be so, particularly as the other variants give much lower distortion than both SE and SE1EL1. In six cases the EL variation gives a slightly worse result than the SE1EL1 variation. One of these cases is due to rounding, however we conjecture that in some graphs, particularly small ones, applying the spring embedder allows the edges to settle slightly more effectively than using the edge length alone.

It is difficult to accurately measure the aesthetics in the graph, so we justify the tuning of the graph from a comprehensibility perspective by giving some example layouts. In some cases the graphs displayed have been chosen from amongst the 10 alternatives to best illustrate the concepts discussed.

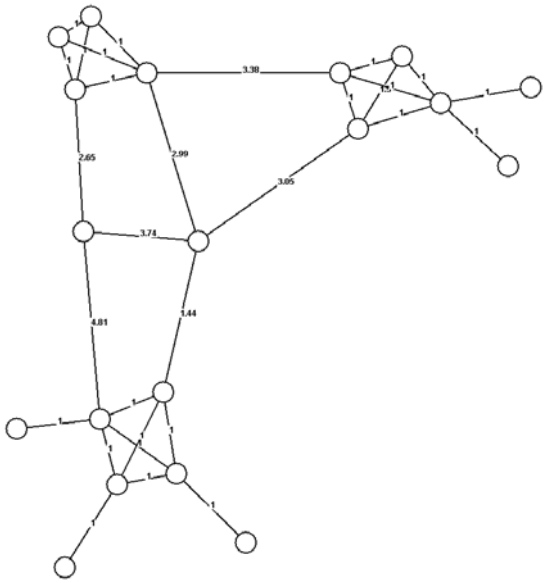

Figure 4a
Graph 'clustered' drawn with the SE variation.

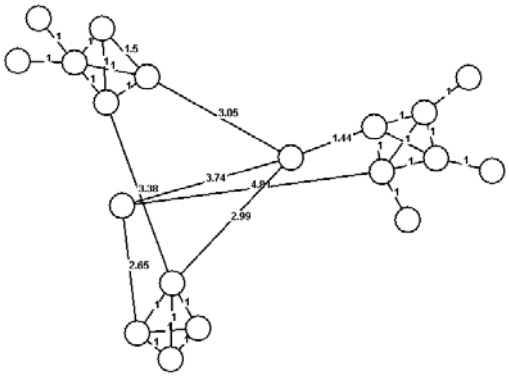

Figure 4b
Graph 'clustered' drawn with the SE5EL1 variation.

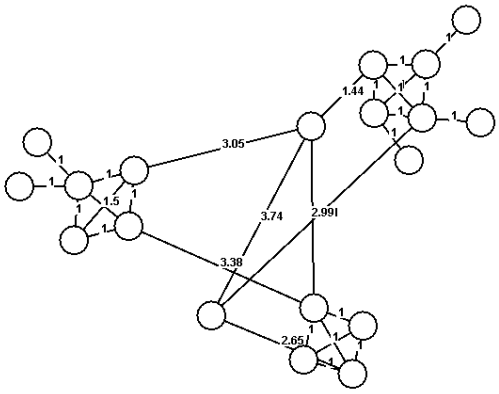

Figure 4c
Graph 'clustered' drawn with the SE1EL1 variation.

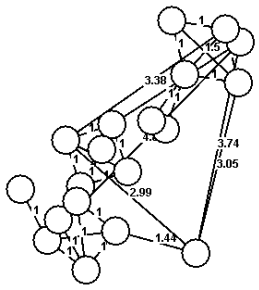

Figure 4d
Graph 'clustered' drawn with the EL variation.

To illustrate the output from the drawing variations, figures 4a, 4b, 4c and 4d show example layouts of the four variations applied to the graph 'clustered'. This graph is a good demonstration, by visual inspection, of the notion that as the edge lengths become closer to the edge weights, the comprehensibility of the graph reduces. Its clear that the layout in Figure 4a, produced by the spring embedder only, variation SE, is very easy to analyse, so that discovering paths between nodes, finding neighbouring nodes and other common graph investigations are relatively unproblematic, but the distortion is 179.27, which is poor compared to the other drawings of the same graph. The first compromise drawing method, shown in Figure 4b, is the SE5EL1 variation, which emphasises the spring embedder. Here, an edge crossing and a certain amount of reduced evenness can be seen. However, the distortion is much better with a value of 17.50. Figure 4c is the compromise variation SE1EL1, emphasising the edge length method. Certainly compared to SE it has worse comprehensibility, and it has more edge crossings than SE5EL1, making manual investigation of the graph harder, but it has a better distortion than both, with a measure of 8.14. Figure 4d is the edge length method only variation, EL. This is a rare example where the edge length only method has a worse distortion, at 8.50, than the SE5EL1 compromise method. This is a very small difference and, as stated above, may be due to some advantage in settling given by the spring embedder for some small graphs.

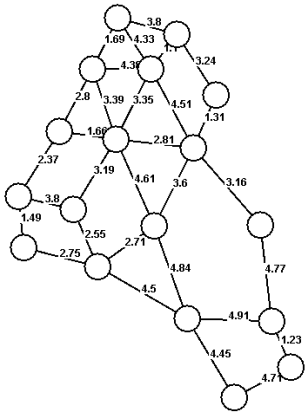

Figure 5a
Graph ‘simple’ drawn with the SE variation.

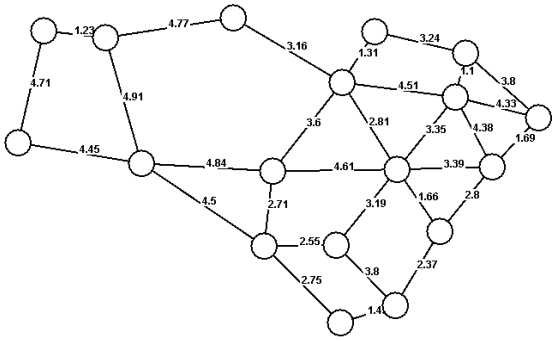

Figure 5b
Graph ‘simple’ drawn with the SE5EL1 variation.

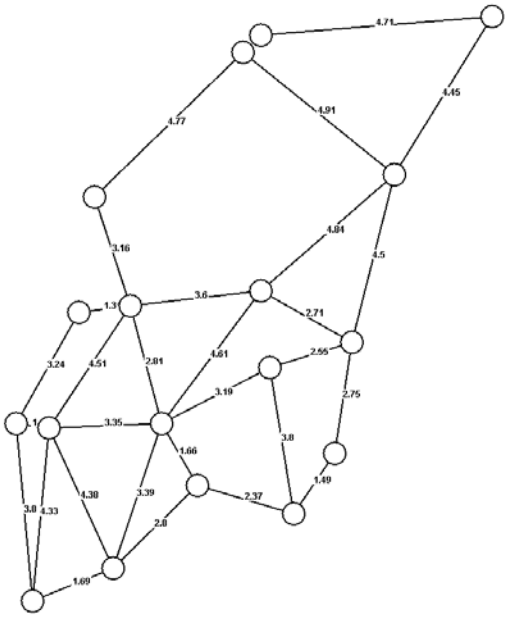

Figure 5c
Graph ‘simple’ drawn with the SE1EL1 variation.

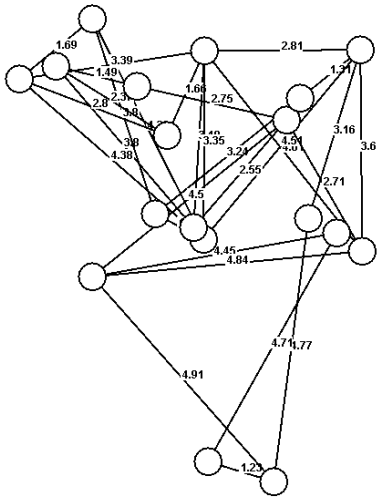

Figure 5d
Graph ‘simple’ drawn with the EL variation.

The example graph ‘simple’ given in Figures 5a to 5d shows a less pronounced difference between the SE and SE5EL1 variations. However, the improvement in distortion is significant, with the SE variation, shown in Figure 5a, having a value of 79.30 and the SE5EL1 variation, shown in Figure 5b, having a value of 56.25. This case indicates that sometimes edge length accuracy can be improved with little negative effect on comprehensibility. The SE1EL1 variation, shown in Figure 5c, has somewhat reduced comprehensibility, but considerably improved distortion, at 17.78. Figure 5d shows the EL only variation, with poor comprehensibility, but minimal distortion of 0.13.

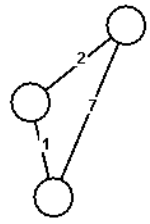

Figure 6a
Graph ‘nonmetric’ drawn with the SE variation.

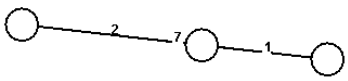

Figure 6b
Graph ‘nonmetric’ drawn with the EL variation. The SE5EL1 and SE1EL1 variations look very similar.

Figures 6a and 6b show layouts of the very small graph ‘nonmetric’, which is a triangle failing the metric inequality, so that it cannot be exactly drawn on the plane with edge length proportional to edge weight. Figure 6a shows a comprehensible drawing, with poor distortion 29821.14, whereas Figure 6b shows an example of the

edge lengths as close as they can be to the ideal, giving a distortion of 20023.50, but the nodes lying in a row are confusing and so adversely affect comprehensibility. The three methods integrating the edge length heuristic produce approximately the same figures, with some minor variation due to rounding.

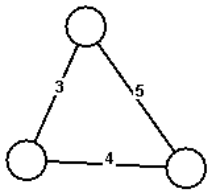

Figure 7a
Graph 'metric' drawn with the SE variation.

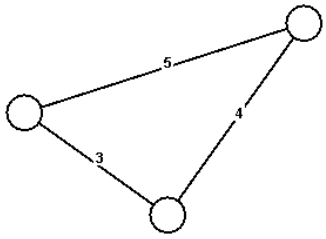

Figure 7b
Graph 'metric' drawn with the EL variation.

Figures 7a and 7b show the graph 'metric', a triangle for which the metric inequality holds. Figure 7a shows the SE layout and has an edge length measure of 2069.17. It is more even than Figure 7b, which is an example layout of the EL variation, but the EL variation has a close to perfect measure of 0.18, which is not exactly zero because of rounding effects.

4. Conclusions

We have introduced a new heuristic for drawing graphs with edge lengths proportional to edge weights. We have shown how it can be combined in a tuneable system alongside the metric spring embedder for drawing graphs according to user preferences.

In terms of compromising edge length proportionality, we feel the notion of tuning the graph for greater or lesser edge length accuracy has been borne out by the results in Section 3. In most cases a more frequent application of the edge length method reduces the edge length distortion, however the relationship appears not to be linear, and more investigation is required.

The notion that applying the edge length method less frequently improves graph comprehensibility is difficult to demonstrate, as the nature of comprehension is not objective. However, from the investigators' personal view of the understandability of graphs drawn with the variations, the hypothesis seems to hold.

Much possible further research results from this preliminary investigation. We have observed that in some cases applying the edge length method after the spring embedder, rather than directly onto a randomly laid out graph, often improved the comprehensibility of the final drawing. This is related to previous research [6], where we found that the spring embedder was made more efficient by a preprocessing step consisting of an earlier version of the edge length method. The connection between the two methods is clearly in need of more research.

Investigation is underway to evaluate the edge length heuristic as a metric embedding method, with no aesthetic consideration for applications such as clustering, against other common methods, in particular multidimensional scaling techniques and hierarchical decomposition.

In order to make the method more usable, implementation optimizations and speed up techniques are required. The software is currently implemented in a way that makes investigation easy, but this comes at a serious compromise to execution time. In addition, to make the system more user friendly, the system should provide an indication of suitable values for various constants used by both the edge length method and the spring embedder. These values could be derived from graph size, density and edge weights.

References

1 P. Eades. A Heuristic for Graph Drawing. Congressus Numerantium 42. pp. 149-60. 1984.

2 S.I. Fabrikant. Visualizing Region and Scale in Semantic Spaces. Proceedings, The 20th International Cartographic Conference, ICC 2001, Beijing, China. pp. 2522-2529. 2001.

3 D. Gilbert, M. Schroeder, J. van Helden. Interactive visualisation and exploration of biological data. Proceedings of 5th Conference on Information Sciences, Atlantic City, USA, February 2000.

4 T. Kamada and S. Kawai. An algorithm for drawing general undirected graphs, Information Processing Letters, v.31 n.1, p.7-15, April 1989.

5 L. Krempel. Visualising Networks with Spring Embedders: Properties and Extensions for Two-Mode and Valued Graphs. 19th International Sunbelt Social Network Conference (Sunbelt XIX). Charleston, South Carolina. February. 1999

6 P.J. Mutton, P.J. Rodgers. Spring Embedder Preprocessing for WWW Visualization. Proceedings of International Symposium on Web Graphics and Visualization, IV02-WGV. pp. 744-749. 2002.

7 Yuval Shavitt and Tomer Tankel. Big-Bang Simulation for embedding network distances in Euclidean space. IEEE INFOCOM 2003, April 2003, San Francisco, CA, USA.

8 C. Walshaw. A Multilevel Algorithm for Force-Directed Graph Drawing. *Graph Drawing*. pp. 171-182. 2000.